

METHOD FOR DETECTING TRANSMISSION CODE ERROR

Publication number: JP9149007

Publication date: 1997-06-06

Inventor: OKUNO MIKIFUMI

Applicant: OKI ELECTRIC IND CO LTD

Classification:

- International: G06F11/10; H03M13/00; H04L1/00; G06F11/10; H03M13/00; H04L1/00; (IPC1-7): H04L1/00; G06F11/10; H03M13/00

- European:

Application number: JP19950304552 19951122

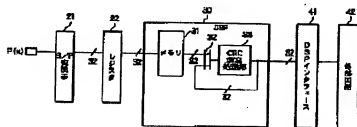
Priority number(s): JP19950304552 19951122

Report a data error here

Abstract of JP9149007

PROBLEM TO BE SOLVED: To conduct cyclic redundancy check(CRC) operation at a high speed in a short time by using a software program of a digital signal processing(DSP).

SOLUTION: A weight required for the CRC operation in a DSP 30 based on an equation of a residue $R(x)$ is stored in advance in a memory 31 in a table form for each bit providing an arithmetic result. The DSP 30 is used, a table in the memory 31 is referenced with respect to transmission data $P(x)$ received in parallel to detect data at an r -bit location required for the CRC operation. The DSP 30 divides the detection result of r -bit by $2^{(r/2)}$, then $r/2$ is substituted to the r and the division is repeated by a number of times of $\log_2(r-1)$ to conduct EXOR by shift of $2^{(r-1)}$ bits for r -number of times with respect to the detection result of r -bit to obtain the residue $R(x)$.



Data supplied from the esp@cenet database - Worldwide

(54) [Title of the Invention]

A METHOD OF DETECTING TRANSMISSION CODE ERRORS

(57) [Abstract]

[Problem to be solved]

To perform a CRC (cyclic redundancy check) operation at a high speed in a short time by using a software program of a DSP (digital signal processor).

[Solution]

A weight required for the CRC operation performing in a DSP 30 based on an equation of a remainder $R(x)$ is stored in advance in a memory 31 in a table form for each bit of an operation result. The r -bit positional data required for the CRC operation for the transmission data $P(x)$ input in parallel is detected using the DSP 30 with reference to the table in the memory 31. The DSP 30 divides the r -bit detection data by $2^{r/2}$ and repeats the division $(\log_2(r)-1)$ times with r taken to be $r/2$ to perform an EXOR operation between the r -bit detection result and 2^r bit shift r times to obtain the remainder $R(x)$.

[Claim for the Patent]

[Claim 1]

A method of detecting transmission code errors by a cyclic redundancy check operation in which a transmission data $P(x)$ expressed by a multinomial is multiplied by x^r (where, r is a positive integer), the $x^r P(x)$ is divided by a generator polynomial $G(x)$ to obtain a r -bit remainder $R(x)$ represented by a multinomial and the remainder $R(x)$ is added to the above $x^r P(x)$ to detect errors during the transmission of information, the method characterized by comprising the steps of:

storing a weight required for performing the cyclic redundancy check operation in a digital signal processor based on the equation of the remainder $R(x)$ derived from the generator polynomial $G(x)$ in advance in a memory in the digital signal processor in the table form for each bit of the operation results;

detecting an r -bit position data required for the cyclic redundancy check operation with respect to the transmission data $P(x)$ using the digital signal processor with reference to the table;

dividing the r -bit detection data by $2^{r/2}$; and

repeating the division $(\log_2(r)-1)$ times with r taken to be $r/2$ to perform an EXOR operation between the r -bit detection result and 2^r bit shift r times to obtain the remainder $R(x)$.

[Detailed Description of the Invention]

[0001]

[Field of the Invention]

The present invention relates to a method of detecting transmission code errors in the subscriber line transmission by a cyclic redundancy check (hereinafter referred to as "CRC") operation by using a digital signal processing device (hereinafter referred to as "DSP").

[0002]

[Conventional Art]

Figure 2 is a drawing describing the CRC which is one of methods of detecting transmission code errors, for example, in the subscriber line transmission. The CRC is one of the methods in which redundant bits are added to a data to detect errors during the transmission of information, and used in the communication protocol in the subscriber line transmission. In the CRC, a transmission data $P(x)$ expressed by a multinomial is multiplied by x^r (where, r is a positive integer) and the $x^r P(x)$ is divided by a generator polynomial to obtain a remainder $R(x)$ of r -bit expressed by a multinomial. An information transmission side adds the obtained remainder $R(x)$ to the $x^r P(x)$ as a check bit and sends it out as transmission information $T(x)$. An information reception side divides the received information $R(x)$ by the generator polynomial $G(x)$ to obtain a remainder. If the remainder is zero, it is regarded that the information is correctly transmitted. If there is a remainder, it is regarded that the transmitted information has an error. A CRC processing which detects such an error has hitherto realized by a hardware because it can be relatively easily realized.

[0003]

Figure 3 is a block diagram of an example of a CRC processor used in a CRC configured by using a conventional hardware. In the CRC processor, when a transmission data $P(x)$ is serially input, it is converted into a parallel data of r -bit by a serial/parallel converter 1 (hereinafter referred to as "S/P converter") and sent to an exclusive-OR (hereinafter referred to as "EXOR") gate 2. A CRC operation circuit 3 is connected to the output of the EXOR gate 2. The output data of the S/P converter 1 and that of the least significant bit (hereinafter referred to as "LSB") of the CRC operation circuit 3 are EXORed by the EXOR gate 2 and then the result is sent to the CRC operation circuit 3. The CRC operation circuit 3 divides the input data $x^r \cdot P(x)$ by the generator polynomial $G(x)$ determined by r to obtain the r -bit remainder $R(x)$. The remainder $R(x)$ is added to the data $x^r \cdot P(x)$ and sent from the transmission side to the reception side as a transmission information $T(x) = x^r \cdot P(x) + R(x)$. A bit width r of the remainder $R(x)$ output from the CRC operation circuit 3 is determined by the information quantity of the transmission data portion $x^r \cdot P(x)$ subjected to the detection of transmission code errors. For example, a value of r in the subscriber line transmission is generally 12 or 32. The CRC operation circuit 3 performs division and is formed by an EXOR logic because the division is executed by, for example, the modulo arithmetic of 2 at which the operation of each digit is EXOR. The CRC operation circuit 3 is configured differently depending upon the generator polynomial $G(x)$. The configuration of a CRC operation with an r of 12 bits (hereinafter referred to as "CRC-12") is shown as a general example in Figure 4.

[0004]

Figure 4 shows an example of configuration of the CRC-12 operation circuit in Figure 3. The CRC-12 operation circuit includes a dividing circuit composed of 12-bit registers 11-1 to 11-12 and EXOR gates 2, 12-1, 12-2, 12-3, ..., and latch circuits 13-1 to 13-12 holding arithmetic results of each bit connected to the output of the dividing circuit. In the CRC processor in Figure 3, the value of the remainder $R(x)$ output as a result of the division of the input transmission data $P(x)$ by the CRC operation circuit 3 is expressed by an EXOR between bit (bit of "1") with a weight represented by the generator polynomial $G(x)$ and an LSB bit in serial CRC operation data. In Figure 4, for the subscriber line transmission, a series of transmission data $P(x)$ is sent in the form of a multiframe. At the time of starting the multiframe, the registers 11-1 to 11-12 are cleared and then the transmission data $P(x)$ is input from the LSB side bit by bit, and all of 12 bits are input by the shift operation of the registers 11-1 to 11-12. The value of the operation result is obtained by performing an EXOR operation between one bit of input transmission data $P(x)$ and LSB of the CRC operation data for all bits (12 times) at a position corresponding to bit with a weight expressed by the generator polynomial $G(x)$. The latch circuits 13-1 to 13-12 are in an enable state at the time of finish of the multiframe and hold each bit of CRC operations as operation results.

[0005]

When the following multiframe transmission data $P(x)$ is input, the CRC-12 operation circuit performs CRC operations in the same manner as the above. The LSB of the operation result is fed back to input side via EXOR gate 2, which means that continuously input transmission

data $P(x)$ are totally subjected to CRC operation. In CRC-12 operation, the results of the CRC-12 operation are output every time transmission data $P(x)$ of 12 bits are input. For example, after transmission data $P(x)$ of 12 bits have been input five times, the CRC-12 operation is performed and an output data of 12 bits are held in the latch circuits 13-1 to 13-12. This means that the value of the remainder $R(x)$ is obtained as a result of CRC-12 operation provided for the 60-bit input transmission data $P(x)$. When the obtained remainder $R(x)$ is represented by a polynomial, $R(x)$ can be expressed by the following equation by taking bit one out of "1" in the output data of the latch circuits 13-1 to 13-12:

$$R(x) = 1 + x^6 + \dots + x^{10}$$

[0006]

[Problems to be Solved by the Invention]

However, a conventional method of detecting transmission code errors performs CRC processing using a CRC operation circuit depending on hardware illustrated in Figures 3 and 4, so that the greater the degree of a generator polynomial $G(x)$ is, the greater the number of circuit elements is, increasing the scale of the circuit. To solve such a problem, it is conceivable that transmission code errors are detected with CRC processing by software program using DSP to downsize the circuit scale. The DSP includes an arithmetic logic section having an arithmetic logic unit (hereinafter referred to as "ALU"), an arithmetic logic operation section with a register, a memory storing data and programs and a microcomputer with input and output sections into which a multiplier is incorporated to accelerate a product-sum operation. Figure 5 shows the process flow of a CRC-32 operation for obtaining $r = 32$ bit remainder $R(x)$ by software program using the DSP.

[0007]

In the CRC-32 operation in Figure 5, when processing is started, the transmission data $P(x)$ is serially input. At step S1, a determination is made as to whether the data of 32 bits have been input. When the data of 32 bits have been input, the process proceeds to step S2. At step S2, a 32-bit input data "crddat" is stored in a memory and called each time a CRC operation result $CRC[i]$ is calculated (32 times in total). At step S2, first, $i = 0$ in the 32-bit input data "crddat" stored in the memory is read and sent to step S3. At step S3, a mask pattern $CRC[i]$ representing a bit position where "1" is set up in the generator polynomial $G(x)$ is generated and sent to step S4. At step S4, the logical AND (hereinafter referred to as "AND") between the input data "crddat" and mask pattern $CRC[i]$ is carried out and the operation result "result[i]" is sent to step S5. At step S5, an LSB of one bit in the CRC operation data and a value stored in the register (or, data of each bit position based on the generator polynomial $G(x)$) are EXORed and the operation result "result 1" is sent to step S6. At step S6, the operation results "result 1" of all of 32 bits are shifted by one bit to the right, and then an LSB of one bit is fed back to the input register at step S7 and the process advances to step S8. At step S8, after one (1) has been added to i , the process proceeds to step S9. A determination is made as to whether CRC processing is conducted 32 times. If a 32-time processing has not been finished, the process returns to step S3 to repeat the above processing. If the 32-time processing has been finished, the CRC-32 processing is terminated.

[0008]

When the CRC-32 processing is executed by DSP, however, processing cannot be performed by a collective bit manipulation, or the processing needs performing bit by bit because data of each bit position based on the generator polynomial $G(x)$ and an LSB of one bit in the CRC operation data require to be EXORed at step S5 and the input data is serial. Furthermore, a break is occurred in the pipeline of DSP associated with the two decision processes at steps S1 and S9, increasing the burden on DSP to significantly decrease an efficiency. In addition, the processes required substantially increase as the degree of the generator polynomial $G(x)$ increases, so that the above burden on DSP causes a problem when CRC processing is carried out within a limited time when a real-time processing such as subscriber line transmission is required. The present invention solves the above problems of the conventional art and provides a method of detecting transmission code errors capable of enabling CRC processing at high speed and in a short time using DSP software program to downsize the circuit scale.

[0009]

[Means for Solving the Problems]

To solve the above problems the present invention provides a method of detecting transmission code errors by a CRC operation in which a transmission data $P(x)$ expressed by a multinomial is multiplied by x^r (where, r is a positive integer), the $x^r \cdot P(x)$ is divided by a generator polynomial $G(x)$ to obtain r -bit remainder $R(x)$ represented by a multinomial and the remainder $R(x)$ is added to the above $x^r \cdot P(x)$ to detect errors during the transmission of information. The following processing is performed in a method of detecting transmission code errors by a CRC operation. A weight required for performing the CRC

operation in a digital signal processor based on the equation of the remainder $R(x)$ derived from the generator polynomial $G(x)$ is stored in advance in a memory in the digital signal processor in the table form for each bit of the operation results. The r -bit position data required for the CRC operation is detected with respect to the transmission data $P(x)$ using the digital signal processor with reference to the table, the r -bit detection result is divided by $2^{r/2}$ and the division is repeated $(\log_2(r)-1)$ times with r taken to be $r/2$ to perform an EXOR operation between the r -bit detection result and 2^r bit shift r times to obtain the remainder $R(x)$. According to the present invention, the remainder $R(x)$ is obtained in the above manner, so that position data required for the CRC operation with respect to the parallel transmission data $P(x)$ is detected using the table in the memory. The detection result and 2^r -bit shift are EXORed i times to calculate the remainder $R(x)$.

[0010]

[Embodiments of the Invention]

Figure 1 shows an example of configuration of a CRC processor used in a method of detecting transmission code errors by CRC operation according to the embodiment of the present invention. The CRC processor is a device for performing CRC-32 processing by DSP, for example, in a subscriber circuit and includes an S/P converter 21 which converts serially input transmission data $P(x)$ into 32-bit parallel data. A register 22 for holding data is connected to the output of the S/P converter 21, and the output of the register 22 is connected to a DSP 30. The DSP 30 includes a memory 31 for holding the 32-bit data output from the register 22, the output of which is connected to one of the input side of an EXOR operation section 32, the output of which is

connected to a CRC processing section 33, the output 32 bits of which is connected to the other of the input side of the EXOR operation section 32 to feed 32 bits back thereto. The CRC processing section 33 is formed by an ALU, multiplier, register, memory and input-output section.

[0011]

A DSP interface 41 is connected to the output of the DSP 30 and an external circuit 42. The DSP interface 41 has a function for controlling communication between the 32-bit signal output from the DSP 30 and the external circuit 42. In the CRC processor in Figure 1, when serial transmission data $P(x)$ is input, the data is converted into 32-bit parallel data by the S/P converter 21 and then held in the register 22. The register 22 is updated at a constant period. The 32-bit data read from the register 22 is held in the memory 31 in the DSP 30 and read by the EXOR operation section 32 when needed. The EXOR operation section 32 performs an EXOR operation between the 32-bit data read from the memory 31 and the 32-bit data output from the CRC processing section 33, and the operation result is sent to the CRC processing section 33. The CRC processing section 33 uses the operation result output from the EXOR operation section 32 as an input data to perform the CRC-32 operation. The result of the 32-bit data obtained by the CRC-32 operation is output to the DSP interface 41. The external circuit 42 assembles the 32-bit data output from the DSP 30 as needed and sent out to a subscriber transmission line in case of a subscriber circuit, for example.

[0012]

Figure 6 is a process flow for the CRC-32 operation using the CRC processor in Figure 1. Figures 7 and 8 show tables for the CRC-32

operation stored in advance in the memory 31 in Figure 1. Figures 9 and 10 show the examples of EXOR processing in the process flow in Figure 6. Referring to Figure 1, Figure 6 and Figures 7 to 10, a description is made of a method of detecting transmission code errors using the CRC processor in Figure 1 according to the present embodiment. When a serial transmission data $P(x)$ is input into the CRC processor in Figure 1, the data is converted into 32-bit parallel data by the S/P converter 21 and then held in the register 22. The tables shown in Figures 7 and 8 need to be stored in advance in the memory 31 in the DSP 30 to perform the CRC-32 processing of the 32-bit parallel data read from the register 22 in parallel by DSP 30. While the data in the table shown in Figures 7 and 8 are shown by HEX (hexadecimal) data on the right end, the data in the process flow in Figure 6 are shown by CRC[i] ($i = 0$ to 31). For example, in a remainder "AMARI_31" on the left end, "1" is set up at 00 bit, 02 bit and 03 bit in 00 bit to 03 bit. This is expressed as D by HEX data. At $G(x)$ bit to $R(x)$ bit, "1" is set up at $G(x)$ bit, this is expressed as one (1) by HEX data. Thus, HEX data of remainder for performing the CRC-32 processing in parallel are stored in the form of a table in advance in the memory 31 in Figure 1.

[0013]

In the process flow in Figure 6, at step S11, the DSP 30 starts processing and determines whether parallel data of 32 bits have been input from the register 22. When the data has been input, the process proceeds to step S12. At step S12, data of bit i is set to 0 and a bit shift quantity "shift" is set to 2^{16} , and the process proceeds to step S13. At step S13, the CRC processing section 33 performs an AND operation between 32-bit tape data "CRC[i]" read from the memory

31 and 32-bit parallel input data "crddat" to detect only bit subjected to the EXOR operation at a subsequent step, thereby masking unwanted bits. The masking result is taken to be "result 1" and the process proceeds to step S14. In the processes subsequent to step S14, EXOR operation is performed between 32-bit data of an output (operation result) with respect to the data converted in parallel and each bit of input data at positions shown in Figures 7 and 8 for each bit. Referring to Figures 9 and 10, a description is made of how the EXOR operation is performed. Figures 9 and 10 show the step of EXOR processing at 31-th bit which is an output obtained when operation is performed by the CRC-32.

[0014]

As shown in Figure 9, at step 13, AND operation is performed between the 32-bit input data "crddat" and the table data "CRC[31]". The 32-bit operation result "result 1" is shifted by 16 bits to the right at step S14 and the shift result "result 2" is sent to step S15. At step S15, an original data "result 1" before it is shifted and the shifted data "result 2" are EXORed, and the operation result "result" is sent to step S16. At step S16, the above operation result "result" is taken to be a first operation result and shifted by eight bits ($16 \times 1/2$) to the right. The process returns to step S14 through step S17 and the data shifted by eight bits to the right and the first operation result "result" are EXORed to taken to be a second operation result "result".

[0015]

As shown in Figure 10, hereinafter processing is conducted in the same manner as described above, the data shifted by four bits to the right and the second operation result "result" are EXORed to be

taken to be a third operation result "result". The data shifted by two bits to the right and the third operation result "result" are EXORed to be taken to be a fourth operation result "result". Furthermore, the data shifted by one bit to the right and the fourth operation result "result" are EXORed to be taken to be a fifth operation result "result". At step S17, when shift quantity "shift" is zero, the process proceeds to steps S18 and S19. When even numbers of positive logic values "1" are included in initially input 32-bit data "result 1", the fifth operation result is "0", and when odd numbers of positive logic values "1" are included, the fifth operation result is "1", which is equivalent to the all bits of the input data "result 1" being EXORed. Thus, 32-bit input data "crddat" shifted by 16, 8, 4, 2 and 1 bits to the right repetitively and the data before shifting are EXORed 32 times respectively through steps 19 and 20, thereby enabling the values of output i bit ($i = 0$ to 31) to be obtained in step S19. The other 29-bit-th to 00-bit-th data shown in Figures 7 and 8 are processed by the same method as described in Figures 9 and 10 to obtain the values of output i bit ($i = 0$ to 31) in the total flow chart. The other 30-bit-th to 00-bit-th data are processed by the same method as described in Figures 9 and 10 and EXORed 32 times in total, thereby terminating a first processing of the CRC-32.

[0016]

Since a subscriber line transmission is used in the present embodiment, the transmission data $P(x)$ is continuously input into the CRC processor on a $P(x)$ frame basis, so that performing the EXOR between the remainder $R(x)$ obtained in DSP 30 by the above procedures and newly input data of 32 bits is equivalent to the CRC-32 operation of continuously input data following previous input data. For example,

if output data is taken from the CRC processing section 33 when the number of inputs reaches five, this means that CRC operation of 160-bit input data is performed. The CRC operation result thus obtained is sent, for example, to the external circuit 42 through the DSP interface 41. The external circuit 42 adds the data of the remainder $R(x)$ from the DSP interface 41 to data $x^r \cdot P(x)$ of transmission data portion and send it to a subscriber transmission line as transmission information $T(x)$. The reception side detects code errors in subscriber transmission line. If the reception side detects abnormality in the subscriber transmission line, it notifies, for example, an upper device of the result. As described above, in the present embodiment, the tables in Figure 7 and 8 showing that bits in what position in input data should be EXORed for each bit of output derived from the generator polynomial $G(x)$ are stored in advance in the memory 31, and the CRC operation is carried out by processing algorithm based on Figures 6, 9 and 10 to obtain the remainder $R(x)$, thus, the present embodiment has the following advantages (a) to (c).

[0017]

(a) A break is hardly occurred in the pipeline of DSP 30 because branch processing is performed once at step S11 in the process flow in Figure 6, enabling CRC operation to be efficiently executed.

(b) The CRC operation of parallel data allows higher speed processing compared with the CRC operation of serial data as shown in Figure 5, enabling a processing time to be significantly shortened.

(c) In the CRC processing, the greater the degree of the generator polynomial $G(x)$ is, the faster the processing is, compared with the CRC operation of serial data, enabling a processing time to be significantly shortened.

Incidentally, the present invention is not limited to the above embodiments, but may be modified into various forms. The following are examples of modifications.

[0018]

(i) In the present embodiment, a description is made of the CRC-32 processing by the DSP 30 in a subscriber circuit. Even if the degree of the generator polynomial $G(x)$ in the CRC operation is increased, the same operational effect as in the above embodiments can be expected.

(ii) The CRC processor in Figure 1 may be differently configured. In addition, the table in Figure 7 stored in the memory 31 may store data in a different form.

(iii) While the present embodiment describes a method of detecting transmission code errors in subscriber line transmission, the present invention can be adapted for a method of detecting transmission code errors except for a subscriber line.

[0019]

[Advantages of the Invention]

As described above, according to the present invention, the weight required for CRC computation is stored in a memory in the table form for each bit of the operation result, the position data of r bit required for the CRC operation is detected referring to the table, and r -bit detection result and x^r bit shift are EXORed r times to obtain the remainder $R(x)$, so that branch processing is few in number in the CRC processing and a break is hardly occurred in the pipeline of DSP, enabling CRC operation to be efficiently executed. Furthermore, the CRC operation of parallel data allows higher speed processing compared with the CRC operation of serial data, enabling a processing time to be significantly shortened. In the CRC processing, the greater the

degree of the generator polynomial $G(x)$ is, the faster the processing is, compared with the serial CRC processing, enabling a processing time to be greatly shortened.

[Brief Description of the Drawings]

[Figure 1]

Figure 1 is a structural drawing of a CRC processor used in a method of detecting transmission code errors illustrating the embodiment of the present invention.

[Figure 2]

Figure 2 is a drawing describing CRC.

[Figure 3]

Figure 3 is a structural drawing of a CRC processor used in a conventional method of detecting transmission code errors.

[Figure 4]

Figure 4 is an example of structure of the CRC-12 processor in Figure 3.

[Figure 5]

Figure 5 is a process flow of CRC-32 operation.

[Figure 6]

Figure 6 is a process flow of the CRC-32 operation in Figure 1.

[Figure 7]

Figure 7 is a table for CRC-32 operation in Figure 1.

[Figure 8]

Figure 8 is a table for CRC-32 operation in Figure 1.

[Figure 9]

Figure 9 is a drawing illustrating an example of EXOR processing in Figure 6.

[Figure 10]

Figure 10 is a drawing illustrating an example of EXOR processing in Figure 6.

[Description of Symbols]

- 21 S/P converter
- 22 Register
- 30 DSP
- 31 Memory
- 32 EXOR operation section
- 33 CRC processing section
- 41 DSP interface
- 42 External circuit

Figure 1

- #1 CRC PROCESSOR OF PRESENT EMBODIMENT OF PRESENT INVENTION
- 21 S/P CONVERTER
- 22 REGISTER
- 31 MEMORY
- 33 CRC PROCESSING SECTION
- 41 DSP INTERFACE
- 42 EXTERNAL CIRCUIT

Figure 2

- #1 TRANSMISSION INFORMATION
- #2 TRANSMISSION DATA $P(x)$ PART
- #3 REMAINDER
- #4 r BIT
- #5 DESCRIPTION OF CRC

Figure 3

- #1 CONVENTIONAL CRC PROCESSOR
- 1 S/P CONVERTER
- 3 CRC OPERATION CIRCUIT

Figure 4

- #1 MULTIFRAME START
- #2 MULTIFRAME TERMINATION
- #3 CRC-12 OPERATION CIRCUIT IN FIGURE 3
- 11-1 REGISTER CLEAR
- 11-2 REGISTER CLEAR
- 11-3 REGISTER CLEAR

11-4 REGISTER CLEAR
11-5 REGISTER CLEAR
11-6 REGISTER CLEAR
11-7 REGISTER CLEAR
11-12 REGISTER CLEAR
13-1 LATCH ENABLE
13-6 LATCH ENABLE
13-7 LATCH ENABLE
13-8 LATCH ENABLE
13-9 LATCH ENABLE
13-10 LATCH ENABLE
13-11 LATCH ENABLE
13-12 LATCH ENABLE

Figure 5

#1 PROCESSING CIRCUIT
#2 * BIT POSITION WHERE "1" IS SET UP IN GENERATOR POLYNOMIAL
#3 TERMINATION OF PROCESSING
#4 PROCESS FLOW OF CRC-32 PROCESSING
S1 INPUT 32-BIT DATA?
S2 STORE "crcdat" INTO MEMORY
S3 MASK PATTERN
S5 EXOR BETWEEN LSB BIT AND VALUE OF INPUT REGISTER
S6 SHIFT BY 32-BIT TOTAL (result 1) BY ONE BIT TO RIGHT
S7 FEED BACK LSB ONE BIT TO INPUT REGISTER

Figure 6

#1 START OF PROCESSING

#2 TERMINATION OF PROCESSING
#3 PROCESS FLOW OF CRC-32 OPERATION IN FIGURE 1
S11 INPUT 32-BIT?
S18 OUTPUT DATA i BIT = result

Figure 7

#1 TABLE FOR CRC-32 OPERATION IN FIGURE 1

Figure 8

#1 TABLE FOR CRC-32 OPERATION IN FIGURE 1

Figure 9

#1 INPUT DATA
#2 AND
#3 OPERATION RESULT
#4 SHIFT BY 16 BITS TO RIGHT
#5 EXOR
#6 FIRST
#7 SECOND
#8 SHIFT BY EIGHT BITS TO RIGHT
#9 DON'T CARE
#10 EXAMPLE OF EXOR PROCESSING IN FIGURE 6

Figure 10

#1 SECOND
#2 THIRD
#3 FOURTH
#4 FIFTH

#5 SHIFT BY FOUR BITS TO RIGHT
#6 EXOR
#7 OPERATION RESULT
#8 SHIFT BY TWO BITS TO RIGHT
#9 SHIFT BY ONE BIT TO RIGHT
#10 ANSWER
#11 EXAMPLE OF EXOR PROCESSING IN FIGURE 6

Drawings of Reference 2 with handwritten reference numbers

(6)

特開平9-149007

10

しておき、DSPを用い、前記テーブルを参照してCRC演算に必要なrビットの位置データを検出し、rビットの検出結果に対してx'のビットシフトによるEXOR演算をr回行って余剰R(x)を求めるようにしたので、CRC演算処理における分岐処理が少なく、DSPにおけるパイプラインが殆ど崩れない。そのため、効率良くCRC演算が行える。さらに、パラレルデータの形でCRC演算を行うので、シリアルでCRC演算を行う場合と比較して高速処理が行え、これによって処理時間を短縮できる。しかも、CRC演算処理において生成多項式G(x)の次数が大きくなればなる程、シリアルにCRC演算処理を行う場合と比較して高速処理が行え、これによって著しく処理時間を短縮できる。

【図面の簡単な説明】

【図1】本発明の実施形態を示す伝送符号誤り検出方法に用いられるCRC演算装置の構成図である。

【図2】CRCの説明図である。

【図3】従来の伝送符号誤り検出方法に用いられるCRC演算装置の構成図である。

【図4】図3中のCRC-12演算回路の構成例を示す*20

*図である。

【図5】CRC-32演算の処理フローを示す図である。

【図6】図1のCRC-32演算の処理フローを示す図である。

【図7】図1のCRC-32演算用テーブルを示す図である。

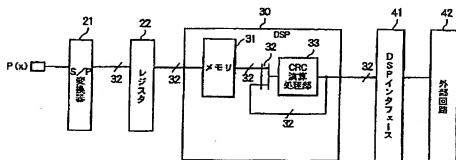
【図8】図1のCRC-32演算用テーブルを示す図である。

【図9】図6のEXOR演算処理例を示す図である。

【図10】図6のEXOR演算処理例を示す図である。【符号の説明】

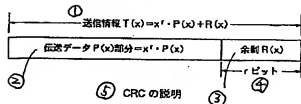
21	S/P変換器
22	レジスタ
30	DSP
31	メモリ
32	EXOR演算部
33	CRC演算処理部
41	DSPインタフェース
42	外部回路

【図1】 FIG.1



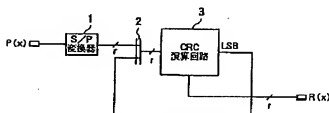
① 本発明の実施形態のCRC演算装置

【図2】 FIG.2



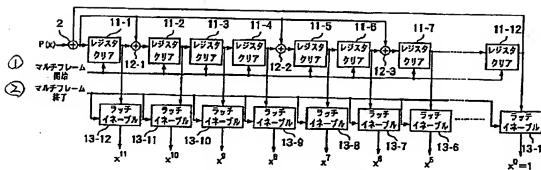
⑤ CRCの説明

【図3】 FIG.3



① 従来のCRC演算装置

【図4】 FIG. 4



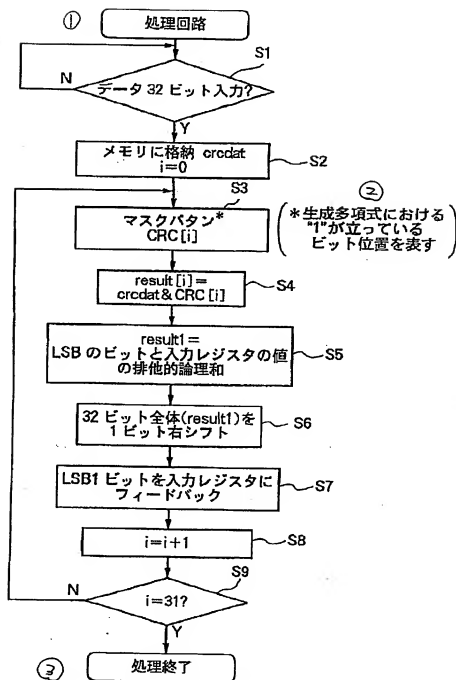
③ 図3中のCRC-12演算回路

【図7】 FIG. 7

[illegible]

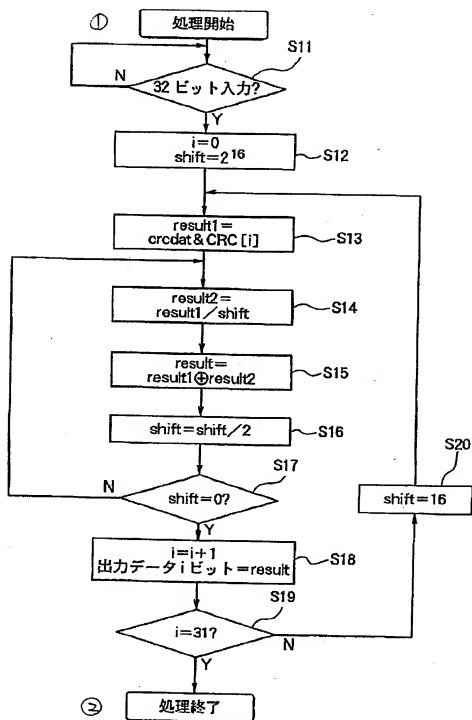
① 図1のCRC-32演算用テーブル

〔図5〕 FIG.5



④ CRC-32 演算の処理フロー

【図6】 FIG.6

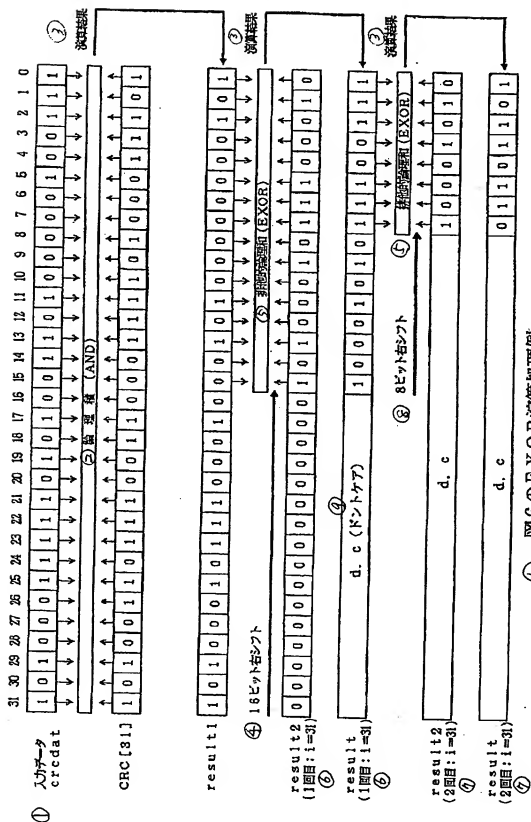


③ 図1のCRC-32演算の処理フロー

【図 8】 FIG. 8

CRC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
AMARI_15	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_14	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_13	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_12	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_11	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_10	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_09	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_08	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_07	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_06	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_05	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_04	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_03	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_02	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_01	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
AMARI_00	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

① 図1のCRC-32演算用テーブル



【図10】 FIG. 10

